Constraint Systems

Scheduling Problems in or-tools

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

For starters, we have (Conditional) Interval variables

```
slv.FixedDurationIntervalVar(start_min, start_max,
                             duration, optional, name)
```

- They are special variables that represent activities
- Each is associated to a start time, end time, and duration
- When we build the var. we specify the range for the start time
- In or-tools, the duration is always fixed…
- …Other solvers support durations as decision variables

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

For starters, we have (Conditional) Interval variables

```
slv.FixedDurationIntervalVar(start_min, start_max,
                             duration, optional, name)
```

The start and end time can be accessed in multiple ways:

- If we just need the ranges:

```
x.StartMin(), x.StartMax(), x.EndMin(), x.EndMax()
```

- If we need to use the values inside a constraint:

```
x.StartExpr(), x.EndExpr()
```

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

For starters, we have (Conditional) Interval variables

```
slv.FixedDurationIntervalVar(start_min, start_max,
                             duration, optional, name)
```

The activities may be optional

- An optional activity may be performed or not performed
- The state is a decision variable (the solver will fix it in a solution)
- Non-performed activities never cause fails
  - E.g. they do not use resources
- They are useful to model planning/resource assignment decisions

We will only deal with non-optional activities

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

Second, we have precedence constraints:

```
x.StartsAfterEnd(y)
x.StartsAfterEndWithDelay(y, delay)
x.EndsAfterStart(y)
...
```

- `x` and `y` are two activity variables
- Each method builds a new constraint

In principle, we could also use:

```
y.EndExpr() <= x.StartExpr()
```

- But then we would need to handle non-performed activities
- Moreover, some resource propagators can use precedences

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

Third, we have resource constraints:

- We have CUMULATIVE, accessible as:

```
slv.Cumulative(X, reqs, cap, name)
```

- `X` is a list of activity variables
- `reqs` is the list of resource requirements
- `cap` is the resource capacity
- There is no need to specify the durations...
- ...Because they are known from the interval variables

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

Third, we have resource constraints:

- For the special case of unary resources:

```
slv.DisjunctiveConstraint(X, name)
```

- Both the capacity and the requirements are 1
- Many propagators have more efficient versions for this case

Scheduling in or-tools

Or-tools provides ingredients to tackle scheduling problems

Finally, we have specialized search strategies:

```
slv.Phase(X, strategy)
```

Where `strategy` can take the values:

- `slv.INTERVAL_SET_TIMES_FORWARD` ("SetTimes" from ch9)
- `slv.INTERVAL_SET_TIMES_BACKWARD` (a "SetTimes" variant)

We do not have a specialized LNS operator

- We need to implement the POS conversion ourselves...
- ...In our example problem, this has been already done

Scheduling in or-tools

An important thing to keep in mind:

Or-tools is not a state-of-the-art scheduling solver!

- Scheduling propagators in or-tools are a bit slow
- Not all the best resource propagators are available
- Only two available search strategies
- No native LNS
- From the python wrapper, we can't choose which propagators to run
  - All propagators are used by default (too slow in most cases)

Other CP solvers (e.g. IBM CP Optimizer) are better

- For real problem, at least use or-tools via C++

Constraint Systems

Minimum Weighted Tardiness

Minimum Weighted Tardiness

Let's consider a simple scheduling problem:

- We have a single, unary, resource
- We need to perform a set of n (non-optional) activities
- Each activity i has a fixed duration d_i
- Each activity has a soft deadline u_i

"Soft" means that the deadline can be exceed, for a cost:

$$c_i = w_i \max(0, s_i + d_i - u_i)$$

- Where w_i is a weight and s_i is the activity start time
- The term $\max(0, s_i + d_i - u_i)$ is called "tardiness"

The objective is to minimize the sum of costs

Minimum Weighted Tardiness

We will tackle the problem via CP and LNS

- This is not necessarily the best choice...
- ...Because the problem is very simple (almost a TSP)...
- ...And because tardiness costs are not CP's forte
- A hybrid CP/LP approach would likely work better

That said, you have a starting script in the start-kit

- The goal is to make improvements (search and LNS in particular)...
- ...And obtain the best possible results on the provided instances